Evolving Living Technologies — Insights from the EvoEvo project

Guillaume Beslon[1], Santiago F. Elena[2,3,4], Paulien Hogeweg[5], Dominique Schneider[6], and Susan Stepney[7]

[1] Université de Lyon, INSA-Lyon, INRIA, LIRIS UMR5205, Beagle Team, Villeurbanne, France
guillaume.beslon@inria.fr
[2] Instituto de Biología Molecular y Celular de Plantas (CSIC-Universitat Politècnica de València), Valencia, Spain
[3] Instituto de Biología Integrativa de Sistemas (CSIC-Universitat de València), Paterna, Valencia, Spain
[4] Santa Fe Institute, Santa Fe, USA
sfelena@ibmcp.upv.es
[5] Theoretical Biology and Bioinformatics Group, Utrecht University, Utrecht, The Netherlands
p.hogeweg@uu.nl
[6] Univ. Grenoble Alpes, CNRS, Grenoble INP, TIMC-IMAG, Grenoble, France
dominique.schneider@univ-grenoble-alpes.fr
[7] Department of Computer Science, and York Cross-disciplinary Centre for Systems Analysis, University of York, Heslington, York, UK
susan.stepney@york.ac.uk

Abstract. The EvoEvo project was a 2013–2017 FP7 European project aiming at developing new evolutionary approaches in information science and producing novel algorithms based on the current understanding of molecular and evolutionary biology, with the ultimate goals of addressing open-ended problems in which the specifications are either unknown or too complicated to express, and of producing software able to operate even in unpredictable, varying conditions. Here we present the main rationals of the EvoEvo project and propose a set of design rules to evolve adaptive software systems.

1 Introduction

Evolution by natural selection is the major source of biological complexity on earth, the origin of all the species we can observe, interact with, or breed. On a smaller scale, evolution is at the heart of the adaptation process for many species, in particular microorganisms (*e.g.* viruses, bacteria or unicellular eukaryotes). Microbial evolution not only results in the emergence of the species itself but also contributes to real-time adaptation of the organisms when facing perturbations or environmental changes. These organisms are not only built up *by* evolution, they are also organized *to* evolve.

As far as information science is concerned, evolution has inspired generations of researchers since the pioneers of the 1950s and 1960s. Nevertheless, most evolutionary algorithms do not take into account the fact that all the molecular systems involved in the evolutionary process are themselves shaped by past evolution. This implies that evolution can influence its own course. This evolution of evolution ("EvoEvo") is at the heart of many phenomena overlooked by evolutionary algorithms, like second-order evolution, evolution of robustness, of evolvability, of mutation operators and rates, co-evolution... Yet, all these processes are at work in living organisms from viruses to whales. Particularly in the case of microorganisms, possibly accelerating their evolution and enabling them to quickly adapt in response to environmental changes like new drugs, pollution, response of their host immune system, or emergence of new ecological niches.

The EvoEvo project was a 2013–2017 FP7 European project aiming at developing new evolutionary approaches in information science and producing novel algorithms based on the current understanding of molecular and evolutionary biology, with the ultimate goals of addressing open-ended problems in which the specifications are either unknown or too complicated to express, and of producing software able to operate even in unpredictable, varying conditions. To do so, the project consortium proposed to start from experimental observations of the evolution of microorganisms under laboratory conditions and to use these observations to reproduce EvoEvo, first in computational models, and then in application software. Our aim was to observe EvoEvo in action, model EvoEvo, understand EvoEvo and, ultimately, reproduce EvoEvo to exploit it in software and computational systems.

In this article we briefly present the rationale of the EvoEvo project. We then focus on the main outcomes of the project, with specific emphasis on those that are likely to have an influence on evolutionary computation and evolutionary software development, as detailed in the project deliverables[1].

2 Overview of the EvoEvo project

2.1 Introduction

One ultimate goal of Information and Communications Technologies (ICT) is to improve human life through the extension of human capacities, abilities or communications. Yet, one of the profound movements that traverse modern social and human sciences is that the world cannot be described as a stable system. Humans and societies continuously change due to the many interactions that lead to instability, to the emergence of new social groups, ideas, modes, media. But today's ICT can barely tackle such highly unstable situations: Every encountered situation needs to be foreseen long before it occurs, at the time the

[1] The following text is massively derived from the EvoEvo project documents. In particular, many paragraphs are derived from the EvoEvo Description of Work (DoW) and from the project Final Report (EvoEvo Deliverable 6.8), available at www.evoevo.eu.

software is designed; and the development cycle of ICT systems is so long that its environment is likely to have changed before the first release of a system. A consequence of this necessary stability of today's ICT systems is that users – individuals and society – must adapt to the ICT systems that are supposed to serve them. Inside the ICT world, the same difficulties are at work since software systems cannot efficiently adapt to the emergence of other pieces of software (or new releases of existing ones) in their environment. Thus, one of the challenges of modern ICT is to develop technologies that are able to adapt dynamically to the evolution of their context, of their user, of the data they receive, and of other systems they interact with – in a single word, of their environment.

The situation in completely different when looking at biology: Evolution, the process that created (and still creates) all the diversity of life, is a process by which generations of organisms continually adapt to their environment. Moreover, the environment of an organism is never stable as it also depends on the evolution of other organisms. Higher eukaryotes have evolved complex sensori-motor systems to adapt their behavior to their changing environment. Microorganisms are less sophisticated systems that lack complex sensori-motor abilities, yet they efficiently use mutation and selection to dynamically adapt to new conditions. Recent experimental evolution results have shown that they are able to evolve at an amazing speed: in virtually all experimental studies that have used bacteria or viruses, important phenotypic innovations have emerged in only a few tens of generations [16]. These results show that, more than being adapted to a specific condition, microorganisms are adapted to evolve: evolution has optimized their own ability to evolve, as a primary means to react to environmental changes. The central idea of the EvoEvo project is that this "evolution of evolution" could offer ICT new paradigms to enable computational systems to dynamically adapt to their environment, *i.e.* to their users, domain of use or condition of use.

2.2 How can evolution evolve?

Variation and selection are the two core engines of Darwinian Evolution. Yet, both are directly regulated by many processes that are themselves products of evolution (*e.g.* DNA repair, mutator genes, transposable elements, horizontal transfer, stochasticity of gene expression, sex, network modularity, niche construction...). Moreover, in a biological system, variation and selection do not act at the same organization level. While variation modifies organisms at the genetic level (by modification of their DNA content), selection acts at the phenotypic level (on the whole organism). The genotype-to-phenotype mapping summarizes in a single conceptual entity (the "mapping") the complex molecular processes by which information flows from the genetic sequence to the organism's phenotype. It captures in a single abstract process different phenomena such as mRNA transcription, gene translation, protein folding, biochemistry and cell dynamics. Again, all these process are themselves dependent on the genetic material that encodes the decoding machinery. Hence the genotype-to-phenotype mapping is itself evolving. Since this mapping directly influences the phenotypic

consequences of a DNA modification, the evolution of the genotype-to-phenotype mapping is likely to change the evolutionary dynamics, *e.g.* by buffering the effect of mutations.

"Evolution of Evolution", or "EvoEvo", encompasses all the processes by which evolution is (or may be) able to influence its own dynamics and to accelerate (or slow down) its own course depending on the environmental conditions. EvoEvo is thus a very general concept that can be difficult to study as a whole, given the wide diversity of mechanisms at stake. That is why, in the context of the EvoEvo project, we decided to focus on four simpler concepts, directly linked to EvoEvo but easier to define, hence to measure *in vivo*, to model *in silico* and, ultimately, to exploit in evolutionary software.

Variability. Variability is the ability to generate new phenotypes, by genetic or epigenetic mutations or by stochastic fluctuations. It is a necessary condition for any evolutionary process to take place. In biological organisms, the amount of variability is controlled by complex pathways that *e.g.* correct DNA mismatches or double strand-breaks. In an ICT context, evolution of variability could help the evolving system to quickly discover new solutions either on a transient or on a stable way through efficient exploration of the functional space. Moreover, in real biological systems, mutational operators are highly diversified: they include not only point mutations, but also large chromosomal rearrangements that can rapidly reshuffle the chromosome organization, extend or reduce the gene repertoire of an organism, or even duplicate its entire genome through whole genome duplication. Current evolutionary algorithms exploit only a tiny part of this complex mutation repertoire.

Robustness. Although necessary, variation is a dangerous process since it produces deleterious mutations that lead to maladapted individuals. Robustness may evolve to correct these deleterious effects. It enables evolving systems to support mutational events without losing fitness, through *e.g.* canalization or the selection of structures that create neutral landscapes. In an ICT context, selection of robustness may favor the emergence of an "organism" structured such that its service will not be perturbed by the random occurrence of mutational events.

Evolvability. Depending on the genotype-to-phenotype mapping, the proportion of deleterious/neutral/favorable mutational events may vary. Evolvability is the ability of a specific genotype-to-phenotype mapping to increase the proportion of favorable events. This can be done by the selection of specific genome structures or by the selection of specific networks structures. In an ICT context, evolvability would enable evolution to exploit past events to increase the system's ability to adapt to new users or conditions.

Open-endedness. Biological evolution is not directed towards a specific target. On the contrary, evolution has the ability to generate new challenges while evolving, *e.g.* by exploiting new niches created by the evolution of other species. In an ICT context, open-endedness [2] can be exploited when an application is made from an ecosystem of evolving individuals. In such

a structure, new functions would arise continuously by emergence of new species in the ecosystem and/or the extinction of maladapted ones.

2.3 A route from biological evolution to artificial evolution

The idea of using a bio-inspired evolutionary metaphor in ICT has led to many powerful developments such as genetic algorithms, evolutionary strategies, and genetic programming. However, most of these developments stay far from current knowledge in evolutionary and molecular biology. The EvoEvo project aimed at creating a true interdisciplinary consortium gathering experimental and computational biologists as well as computer scientists. This gives rise to a difficult question: how can one guarantee that the biological foundations of EvoEvo – which we aimed to observe *in vivo* – are effectively and efficiently transfered to the ICT world and to the computational application? To tackle this issue, we proposed a particular route from evolutionary biology to artificial evolution through modeling. The project was organized to benefit from the pivotal role of computational modeling of evolution, aka *in silico* experimental evolution [16, 3]. These models are computational artifacts that mimic the phenomenon observed *in vivo*. They thus constitute an intermediate step between life science and ICT. Now, models must not be mistaken for application code. Their objectives are – and must stay – clearly different. That is why the transition from life science to application code was organized in two steps. The *in vivo* experiments were modeled *in silico*, and those models were then reinterpreted to develop a computational framework that benefited from them but enabled the introduction of simplification and/or generalization of the full model's bio-like structures.

2.4 EvoEvo... What for?

As explained above, the EvoEvo project covered a large range of research domains, from experimental biology to software development. To ensure that EvoEvo would produce results that fulfill the target of the EVLIT European program[2] (designing "empirical, theoretical and synthetic approaches that define the key bio-inspired principles that can drive future living technologies and the environment to use them in a controlled way"), we proposed to develop proof-of-concept applications aiming to demonstrate the power of EvoEvo. This opened a difficult discussion within the consortium on which kind of application were (1) doable in a reasonable time and (2) likely to benefit from the EvoEvo principles. We finally decided to develop "personal evolutionary companions": software systems that continuously evolve through the interaction with their user by means of sensor networks. To keep the system's complexity low enough, we focused on a very specific situation: the interaction between a dancer and the music. This led to the development of the EvoMove system, a personal companion that learned to play music while the dancer is dancing through continuous adaptation to the dancer's moves. EvoMove is briefly described in section 4.2.

[2] ICT-2013.9.6 – FET Proactive: Evolving Living Technologies (EVLIT).

3 Results: EvoEvo insights from biological and *in silico* evolutionary experiments

3.1 What is evolution?

After Darwin and "The Origin of Species" (1859), evolution can be basically defined by the process of species emergence through the simultaneous action of variation and selection. Given that evolution takes place in the context of populations, a third mechanism was later added by Motoo Kimura [20]: Neutral genetic drift, which accounts for the unavoidable effect of sampling in finite populations. This mechanistic definition of evolution can easily be shared by different disciplines, from evolutionary biology to computer sciences, even though the underlying mechanisms can be very different. However, the disciplines are much less in agreement when defining evolution by its consequences on the organisms and on the species. While computer scientists and mathematicians tends to consider that evolution is an optimization process, this notion of optimization is not clearly defined in biology, not least because there is no universal definition of an "optimum" and of what must be considered as the "fitness" of the organisms (*i.e.* their reproductive success). When dealing with a "simple" artificial system, the fitness can often be easily computed though a predefined algorithm. But this is not the case for a real organism in which the fitness encompass many different effects (*e.g.* number of offspring, offspring viability, sexual selection...). If Darwinian evolution is an optimization process, we are completely blind to what it optimizes.

During the EvoEvo project, we developed many computational models of evolution with different formalisms [9, 28, 31]. Hence, the definition of fitness in these different models also varied. However, a strong point of convergence of all these models is that the measured fitness is generally different from the specified fitness (*i.e.* from the coded criteria of success at the individual level). In these models, which all encoded an optimization process through selection at the individual level, the evolutionary outcome was a much more complicated process that involved different levels of organization: the interaction between the multiple levels of organization encoded in the models (genotype, phenotype, population...) blurred the optimization criterion that acted at a single level (the phenotype). Hence, we were facing the same kind of problem biologists face when measuring the evolutionary success of an organism. On the one hand, this is an interesting result for biology (as it enabled us to identify new evolutionary mechanisms) but on the other hand, it makes it difficult to transfer our models and results to the computational world since the outcome of evolution is no longer clearly defined.

To overcome this difficulty, we propose an alternative definition of the evolutionary outcome. Even if evolution is not optimization, it can still be defined as the process by which an organism (or a species) accumulates information about its environment while thriving in it. This definition opens interesting issues. Information is evidently accumulated in the inherited material (the genome), but it can also be accumulated in other characteristics of the organisms, providing

these characteristics can be transmitted to the next generation. An example of this process is the spatial or temporal organization of the population (section 3.2). The information accumulation in the genome depends on many parameters that are likely to vary during evolution (the genome size, the coding proportion, the genome structure, the epigeneetic methylation patterns) and this opens a clear path to EvoEvo mechanisms (section 3.3). Viewing evolution as information accumulation also poses the question of information stability: information can accumulate to the extent that the organisms are able to transmit it efficiently to the next generation. This directly links the notion of fitness to the notion of robustness and evolvability (section 3.4).

3.2 Long-term information integration

Long-term information integration – the capacity of evolving systems to accumulate information over long time scales, possibly overriding immediate disadvantages – has long been taboo for explaining what has evolved, because without explicit modeling it invites just-so stories. Moreover, classical evolutionary models and algorithms in which evolution is limited to modifying allele frequencies, or few parameters, do not allow long-term information integration.

All the models we designed or used during the EvoEvo project included not only a genetic information level but also many additional degrees of freedom. These degrees of freedom were different in the various models, but included: the spatial position of the organisms (and subsequently the spatial interaction between the organisms); non-coding sequences and relative position of the genes along the genomic sequence; waste production and release in a shared environment... In virtually all *in silico* experiments we conducted with these models, evolution found ways to use these degrees of freedom to accumulate information about its environment. This information was then empowered to regulate evolution (*e.g.* in the case of additional degrees of freedom at the genomic sequence level, see next section) or led to transitions from individual levels of selection to higher level selection of composite entities, one of the "major evolutionary transition" [30] and a clear stepping-stone for open-ended evolution [2].

One example is the *in silico* experiments of E. S. Colizzi at Utrecht University [9], which shows how such higher level selection can overcome the well known tragedy of the commons problem. When the production of an essential "common good" is costly, individual level selection will reduce its production leading to eventual extinction of the whole population. However when the population is embedded in space this is no more the case. Instead, when the cost of production is high enough, the population splits in a subpopulation of cheaters, not producing the common good, and a subpopulation that produces much of it. But space enables both subpopulations to self-organize, hence evolving stable higher-level entities: both subpopulations together form traveling waves, the producers in front, and the cheaters in the tail (figure 1). It is this robust spatial organization which overcomes individual level selection to avoid the cost, and thus prevents the tragedy of the commons. Strikingly the system as a whole will produce more common good the higher the cost! Interestingly, the structure of the traveling

waves is not encoded in the genome of the organisms. It is rather encoded in the dynamic eco-evolutionary interactions between producers and cheaters, *i.e.* at a higher organization level.

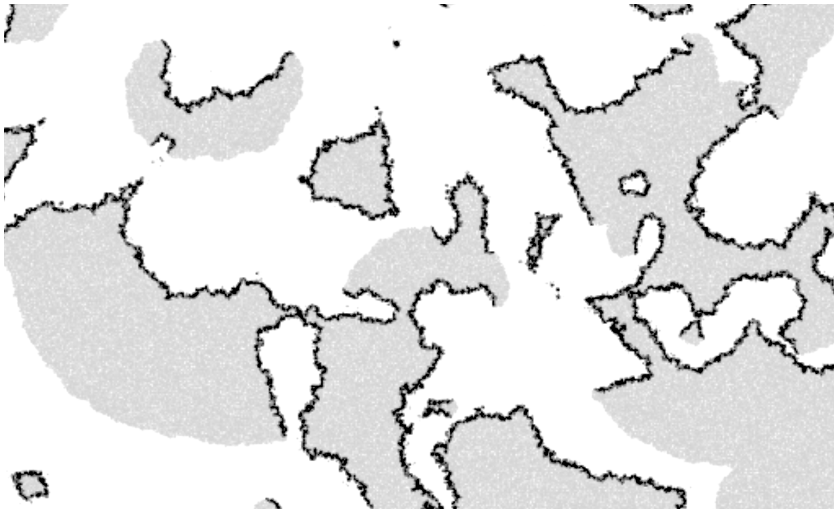

Fig. 1. Example of the high-level structures (traveling waves) created by the interaction between producers (gray) and cheaters (black). The producers form an expanding front that colonizes free space. The cheaters follow common good gradient, hence the front, thereby leading to the extinction of the wave. This frees space that hence become available for another wave. This interaction between front expansion and space freeing results in a stable high-level spatial temporal dynamics which enables evolution of high production common good despite high cost to individuals (reproduced from [9]).

The "real world" (in our case, the biotic world) contains many of such degrees of freedom (*e.g.* spatial interactions, non-specific interactions between molecular compounds, non-genetic inheritance) and evolution can empower them to accumulate information. Our results suggest that including similar "messy" interactions between all levels and components could be a way to increase the innovation power (the "open-endedness") of evolutionary algorithms, although maybe not their optimization efficiency. A first, easy, step would be to embed populations in space (providing individuals can spatially interact with each others), as it improves even simple evolutionary search. Similarly, exploiting long term information integration by using sparse fitness evaluation, in which subproblems co-evolve with solutions, could improve evolutionary search [11, 22].

3.3 Evolution of genetic architecture and the role of non-coding sequences

One fundamental mechanism we identified is the evolution of the size and the structure of the genome. The size and structure determines the dimensionality

of the evolutionary search space and the overall mutation pressure; it also determines the relative pressure due to the different mutational operators and the kind of genomic changes these mutational operators can achieve.

While most evolutionary theory in biology, as well as in computer science, has focused on point mutations and crossovers, we have highlighted the role of mutations which change the size of the genome – *e.g.* large and small duplications and deletions – thereby changing the dimensionality of the fitness landscape and search space. We have shown that this increases the effectiveness of evolutionary search in several ways. Typically, successful evolutionary adaptation involves early genome expansion, followed by streamlining, whereas in the absence of genome expansion less fitness is obtained in the end: the gradual reduction of the dimensionality of the search space facilitates optimization [10]. These operators hence increase the evolutionary potential: by allowing the information content of the genome to evolve, they facilitate adaptation to changing conditions. We should note however they also impose strong robustness constraints on the genome size, hence bounding the quantity of information the genome can accumulate [13].

Structuring of genomes goes beyond the effect on genome size and so does its impact on evolution. Indeed, the variation operators modifying the genetic content are differently impacted by the genome structure (*e.g.* including non-coding sequence between two genes does not change the effect of point mutations while it strongly changes the effect of duplications). Hence, an evolvable genetic structure enables evolution to fine tune the distribution of offspring fitness, selecting for robustness and/or evolvability when needed. This fine tuning can even allow for simultaneous selection of robustness *and* evolvability, as exemplified by the results of Jaap Rutten [29]. His experiments show that organisms in which the point mutation rate is raised by a factor of 100 react by reorganizing their genome. While theoretical results with simpler models predict a fitness loss due to the loss of robustness, genome reorganization enables the organisms to change the distribution of offspring fitness, and so to increase evolvability. Although the high mutation rate occasionally leads to fitness loss through loss of the master sequence, the increased evolvability enables the population to quickly recover, hence keeping the fitness of the best individuals at the same level (and sometimes at a higher level) than that of the wild-type individuals that evolved under a constant mutation rate. Similar results have been obtained in another class of model studied in the course of the project: by evolving RNA sequences, E. S. Colizzi [8] has shown that the RNA structure (the equivalent of the genome structure in the RNA model) is selected such that the population contains an efficient proportion of mutated sequences and such that the mutants help the master sequence to thrive.

In conclusion, genome structuring, mediated by a plethora of mutational operators, is a powerful mechanism for EvoEvo. It helps explain recent observations of very fast adaptation to novel environments in experimental evolution [27], and may help regulate evolutionary dynamics by changing the impact of

the different kinds of mutation, hence the distribution of sequences/fitnesses in the population.

3.4 On the importance of long jumps

One of the difficult open questions in evolutionary biology (and so far an unsolved issue in artificial evolution) is the question of evolutionary innovation: When trapped on a local optimum, how does a population escape to find a new, higher, peak in the fitness landscape? Different hypotheses have been proposed in the literature, one of the most popular being exploration of the so-called "neutral landscape". In this view, a local optimum can be changed into a plateau when the number of dimensions increases, and this plateau is likely to be connected to higher peaks, whereas in lower dimensions (a shorter genome), it would have been surrounded by fitness valleys. Though this theory has many advocates, it suffers from a major drawback: the curse of dimensionality. When the number of dimensions increases, the time needed to explore the plateau increases exponentially [6], making it very unlikely to find an escape route in a reasonable time. Moreover, as we explain above, increasing the size of the genome may have a strong effect on robustness [13], hence limiting the interest of this strategy.

We studied evolutionary innovation in the Aevol *in silico* experimental evolution platform (www.aevol.fr) by evolving populations for a very long time. Once these populations get stuck on local fitness optima, we cloned them and resumed the evolution of the clones. This procedure enabled us to isolate the clones that innovate from those that stay stuck on the initial optimum, and so to analyze the route to innovation [4]. The results emphasize again the role of large scale modifications of the genome structure: In a large majority of the "innovator clones" innovation was triggered by a specific mutational event that strongly increased the evolvability of the clone by increasing the size of its genome, generally through duplication of a small sequence. This result sheds new light on the innovation dynamics: rather than randomly diffusing on a neutral landscape or accumulating deleterious mutations to cross a fitness valley, the innovator clones stay on the top of their local fitness peak but try "long jumps": mutations that directly connect them to a distant part of the fitness landscape. Of course, the vast majority of these jumps are deleterious, but their combinatorics is much larger than the one of point mutations: the number of possible point mutations in a genome is proportional to N, the size of the genome, while the number of possible sequence duplications is proportional to N^3. Hence, a population can quickly explore the whole set of available point mutations (if the population is large enough), ending stuck on a local optimum. Exploring the whole set of chromosomal duplications does takes time, but leads to innovation.

Following this result, a strong recommendation for evolutionary algorithms is to include a set of mutations with a very high combinatorics, that enables long jumps in the fitness landscape. In our experiments this role is played by chromosomal duplications, but other operators could play the same role. One could for instance consider Horizontal Gene Transfer (HGT), providing the source and destination of the gene are different enough (this is not the case in the classical

crossover operators used in evolutionary computation) or large scale modification of the genome conformation and folding.

4 Results: EvoEvo algorithms and applications for living technologies

4.1 EvoEvo algorithms and computational concepts

Based on the insights gained from studying various evolutionary phenomena (section 3), the EvoEvo project developed several novel evo-evolutionary algorithms and architectures.

EvoMachina. EvoMachina [19] is a novel meta-evolutionary algorithm that incorporates several of key findings: that *genomic reorganization* is an important factor in the evolution of evolvability; that the machinery of evolution (expression, replication, etc) is implemented by *machines* that are themselves encoded on the genome, and hence are themselves subject to evolution; that spatial organization of replicating entities provides an extra level of information integration.

EvoMachina allows multiple different types of genomes, allowing appropriate representations and machinery to be used for different parts of the application. For example, the mutation machinery can be encoded in a separate genome, allowing the mutation operators to evolve in a different manner, and at a different rate, from the applications candidate solutions.

An implementation of EvoMachina is available as an open-source Java framework at github/evoevo-york/evomachina. The framework includes a variety of evolutionary variants such as classic EA and microbial GA, as well as the EvoMachina specific operators, and a variety of spatial options, including a well-mixed option and a 2D toroidal grid.

Bio-reflective architecture. We have argued that computational reflection is an essential component of computational novelty generation [2]. Based on this, we developed a new *bio-reflective architecture* [14]. It is a synthesis of concepts from: von Neumann's Universal Constructor Architecture; procedural computational reflection; evolutionary algorithms; computational open-ended novelty mechanisms; the EvoMachina architecture of evolvable active machines and passive genomic structures.

Parts of this architecture were realised in the stringmol automata chemistry and used to demonstrates a form of semantic closure [7]. Parts were realised in a stand-alone evolutionary music application [15], and also informed the "commensual architecture" of the dance application [1] (section 4.1).

The commensal architecture. One of the core universal properties of living beings is their autonomy. Even if some forms of cooperation or altruism can be

observed in nature, every biological system is fundamentally selfish and cooperation can emerge only when multiple levels co-evolve, the selfishness of some constraining the cooperativeness to others. On the opposite, one of the core universal properties of technology is its controllability.

These two antagonistic properties immediately conflict when one wants to design "living technologies". They also conflict when one wants to design open-ended technologies: if open-ended systems are to continuously produce novelty [2], how can they be designed? So when designing living technologies, one of the central problems is to design a system that is autonomous enough to surprise its user (by producing novelties) and, at the same time, is constrained enough to serve the goals it has been built for (as a technology). Since the very beginning of this project, this tension has been at the heart of EvoEvo: if autonomy is one of the core properties of life, how can a technology be simultaneously alive and controllable?

As said above, biological systems can be cooperative or altruistic provided they are embedded in higher/lower levels of evolution that constrain them. We propose here a bio-inspired approach to resolve the autonomy *vs.* controllability conundrum. We called this approach "commensal computation" [1]. In biology a commensal (from the Latin *cum mensa*, at the same table) interaction is a form of mutualism between two organisms where the association is not detrimental but not obviously beneficial to the partners [17]. Indeed, the idea of commensal computation is based on one of the main functions of the gut microbiota: nutrient processing. Gut microbes degrade ingested substances that would otherwise be non-digestible or even harmful to the gut [18]. This role enables the organism to uptake nutrients originating from a wider variety of sources than would otherwise be the case: microbes preprocess the complex flow of nutrients and transfer the results to the host, helping it to regulate its feeding and to extract specific nutrients. While doing so, the microbiota live their own lives, and change and evolve according to their environment: what the host eats: The commensal association of the microbiota and the host contains a part of autonomy (the microbes) and a part of control (the host).

We propose to organize living computational system following the manner in which host and microbiota are engaged in a mutualistic association. In commensal computation, the complex data (*e.g.*, data generated by the sensor networks) are pre-processed by a virtual microbiome that transforms them in digestible data that the processing system can use. Such an architecture differs from classical pre-processing in that here the pre-processing is performed by an evolving community of virtual bacteria that uptake data, transform them in recognizable objects (symbols, clusters, classes, ...) and feed them to the main processing system. In the context of the EvoEvo project, we used a subspace-clustering layer to implement the commensal level[3]: virtual bacteria evolve subspace classifiers

[3] Clustering is a data-mining task that aims to group objects sharing similar characteristics into a same cluster over the whole data space. Subspace clustering similarly aims at identifying groups of similar objects, but it also aims at detecting the subspaces where similarity occurs. Hence it can be conceived as "similarity examined

and send the result to the processing layer. The interest of subspace classification here is that it enables a sensor network (or more generally the source of data) to change its dimensionality (*e.g.*, adding/removing sensors) without causing a complete failure of the classification: new dimensions can be dynamically added to the system and will (or will not) be integrated to the clustering depending on their pertinence with regards to the existing clusters and to the data.

We designed an evolutionary subspace clustering algorithm using the evolutionary principles detailed in the previous sections. In particular, we tried to empower the principle of an evolvable genomic structure (variable number of genes, regulation of coding proportion...) and the principle of using a large variety of mutational operators (point mutations, gene duplication and deletion...). Simultaneously we tried to simplify as much as possible the models that were used as a source of inspiration in order to reduce the computational load and to enable real-time execution of the algorithm, a mandatory property for its use in an evolving personal companion.

These principles led to a series of algorithms from "Chameleoclust" [24] to "SubCMedian" [26]. All these algorithms have been tested on public benchmarks and have shown state-of-the-art levels of performances.

4.2 Proof of concept: Evolving a living personal companion

One of the objectives of EvoEvo was to produce not only concepts but to test these concepts in proof-of-concept applications. This has been done in two steps corresponding to the commensal architecture described previously. In a first step, we designed the "commensals": artificial entities able to evolve in an environment composed of static and dynamic data (the evolutionart subspace clustering algorithms described above). Then, in a second step, we used these algorithms as a commensal pre-treatment layer in a musical personal companion: EvoMove.

The ultimate proof-of-concept of our EvoEvo approach of evolving software was to evolve a real application and to have it used by a real "naive" user. That is why we choose to implement a personal companion, software able to continuously evolve through interaction with its user. Then, in order both to address naive users and to test the software in short training sessions, we decided to design a musical personal companion: a system that would be able to evolve music depending on the performance of a dancer and that would evolve in real time while the performance is ongoing. This resulted in the EvoMove System [25].

The principles of EvoMove are detailed in figure 2. The system leverages the evolutionary subspace clustering algorithm described above, by embedding it into a commensal architecture: the moves of the dancer are captured through Inertial Measurement Units (IMU) and transmitted to the subspace clustering algorithm that identifies moves similar to those it has seen before. The subspace clustering is then computed in complete autonomy, intentionless and without any

under different representations" [23]. Subspace clustering is recognized as a more complicated and general task than standard clustering. Moreover, retrieving meaningful subspaces is particularly useful when dealing with high dimensional data [21].

need for calibration. The identified clusters are then transmitted to the "host", here a sound generating system that triggers new sounds each time a new cluster is identified and that repeats this sound each time this cluster is activated again. The commensal architecture hence results in a host fed by motion data and producing music, and a bacterial community that processes the motion data, helping the host to interpret the moves. Both organisms thus "eat at the same table" (the motion) and co-evolve. The music produced by the host depends on the command objects produced by the virtual bacteria. The motion fed to the bacteria depends on the movements the users make in reaction to the music they hear.

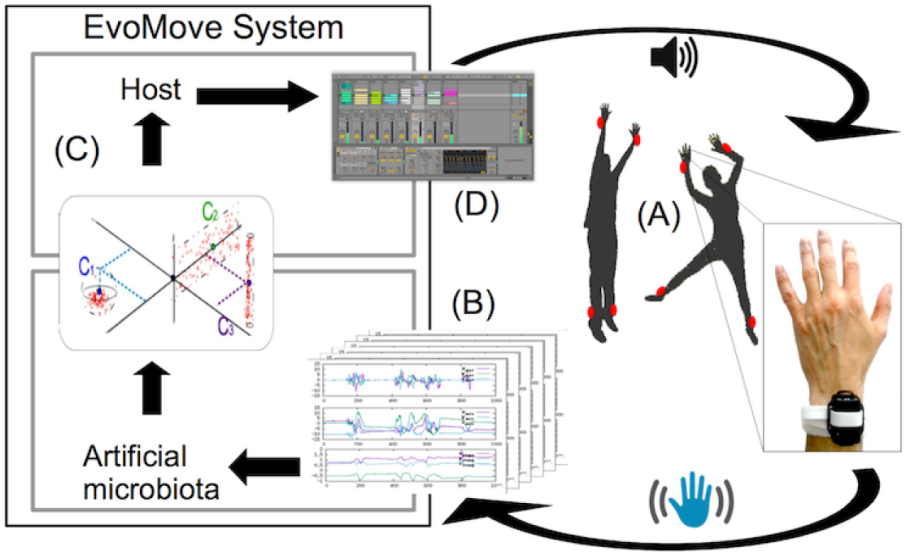

Fig. 2. The EvoMove feed-back loop. (A) Dancer moves are captured by Inertial Measurement Unit (IMU). (B) The sensors produce a high-dimensional data-stream. (C) This data-stream is clustered by SubCMedian algorithm that outputs a set of clusters. (D) The sound system outputs sounds that are immediately perceived by the dancers who can adapt their dance, leading to reciprocal adaptation of the clusters, hence of the music. This feedback loop produces coherent music due to the close integration of dancers and clustering algorithm: the duration of the loop is less than 1 second, enabling real-time response of the system.

Thus, this system creates a feedback loop including the human user. One iteration of the loop is run approximately every second. This timing is short enough to allow interaction. Contrary to most software where the human is acting on a system, here the users are acting *in* the system. They do not have full freedom about what sounds will be produced, but they can influence them.

They have to decide how they react to what could be called "sound proposals" from the system, and this decision changes the shape of what the system produces next. And contrary to most of music software, the output of the system is not only the sound produced, but what is produced at each step of the loop and especially what is visible: music and moves.

[12] presents a short video of an EvoMove test with a EvoMove-naive dancer (an experienced dancer who had not used the system before and who did not know its mechanisms). EvoMove has also been used during the "Meute" dance performance, which has been publicly presented at several dance festivals in Lyon (France). It is difficult to claim that such a system "works" or not since it is strongly dependent on the sensations of the user. But in all these situations, EvoMove has convinced the dancers by stimulating them in such a way that they were all eager to use it again and interact with it on other occasions. Our supposition about what produces this, besides the possibilities offered by the commensal layer, is the integration of the human user in the feedback loop. As a consequence, the dancers are always adapting their own moves and actions to fit what they understand of the state of the system. Thus, even though the machine part of the system is deviating from what would be seen as interaction, the human is able to follow it so as to keep this interaction alive. This process does not have to be conscious from the user perspective. Just by investing effort into being understood by the system, the user adapts their actions alongside the system state changes. Hence, the "living technology" is not (or not only) in the system; it is rather in the close interaction of the system and its user, both reacting to each other's proposal.

5 Conclusion

As the 2011 FET Consultation Report "Living Technology, Artificial Systems, Embodied Evolution" shows, many approaches have been proposed to create living technologies. Now at the end of the EvoEvo project, and having created what we think is a living technology ("EvoMove"), one can draw the big picture of the design principles we identified and briefly exposed here. Indeed, we claim that the key insight into building evolutionary living technologies is to go back to a fundamental property of living systems. Living systems are in essence strongly integrated systems, while technological systems are, by construction, strongly modular systems. Evolutionary living technologies will only be efficient if they are strongly integrated within the systems (in order to enable the system to innovate) and with their users, be it a real person (as in EvoMove), or a software entity (as in commensal architecture).

In some sense, this proposal is not a total surprise, since it is a similar mind shift as the one that happened at the end of the 1980s in robotics. The development of Behavior-Based Robotics under the impulsion of Rodney Brooks was nothing else than the close integration of robots with their environments and of robots components one with the others [5]. We now propose that software systems themselves, although they are not physical entities, follow the same path

in order to be able to leverage Darwinian evolution to dynamically react and adapt to their users. This will enable living software systems to co-construct their behavior with a user who, in that same moment, will become a partner of this behavior.

Acknowledgments

This work was supported by the European Commission 7^{th} Framework Program (FP7-ICT-2013.9.6 FET Proactive: Evolving Living Technologies) EvoEvo project (ICT- 610427, http://www.evovo.eu/). The authors thank all the partners of the EvoEvo project for fruitful discussions.

References

1. Abernot J., Beslon G., Hickinbotham S., Peignier S., Rigotti, C. (2017) Evolving Instrument Based on Symbiont-Host Metaphor: a Commensal Computation. *Journal of Creative Music Systems*, 2(1):1–10.
2. Banzhaf W., Baumgaertner B., Beslon G., Doursat R., Foster J. A., McMullin B., Veloso de Melo V., Miconi T., Spector L., Stepney S., White R. (2016) Defining and Simulating Open-Ended Novelty: Requirements, Guidelines, and Challenges. *Theory in Biosciences*, 135(3):131–161
3. Batut B., Parsons D.P., Fischer S., Beslon G., Knibbe C. (2013) In silico experimental evolution: a tool to test evolutionary scenarios. *BMC bioinformatics*, 14(15):S11
4. Beslon G., Liard V., Elena S.F. (2016) Evolvability drives innovation in viral genomes. In:*2nd EvoEvo Workshop, satellite workshop of CCS2016, Amsterdam, September 2016*, 6 p.
5. Brooks R.A., (1990) Elephants don't play chess. *Robotics and autonomous systems*, 6(1-2):3–15.
6. Chatterjee K., Pavlogiannis A., Adlam B., Nowak M.A. (2014) The time scale of evolutionary innovation. *PLoS computational biology*, 10(9):e1003818
7. Clark E.B., Hickinbotham S.J., Stepney S. (2017) Semantic closure demonstrated by the evolution of a universal constructor architecture in an artificial chemistry *Journal of the Royal Society Interface*, 14:20161033
8. Colizzi E.S., Hogeweg P. (2014) Evolution of functional diversification within quasispecies. *Genome biology and evolution*, 6(8):1990–2007
9. Colizzi E.S., Hogeweg, P. (2016) High cost enhances cooperation through the interplay between evolution and self-organisation. *BMC evolutionary biology*, 16(1):31
10. Cuypers T.D., Hogeweg P. (2012) Virtual genomes in flux: an interplay of neutrality and adaptability explains genome expansion and streamlining. *Genome Biology and Evolution*, 4(3):212–229
11. de Boer F.K., Hogeweg P. (2012) Co-evolution and ecosystem based problem solving. *Ecological informatics* 9:47–58
12. https://youtu.be/p_eJFiQfW1E
13. Fischer S., Bernard S., Beslon G., Knibbe C. (2014) A model for genome size evolution. *Bulletin of mathematical biology*, 76(9): 2249–2291
14. Hickinbotham S., Stepney S. (2016) Bio-reflective architectures for evolutionary innovation. *ALife 2016, Cancun, Mexico*, pp.192–199. MIT Press.

15. Hickinbotham S., Stepney S. (2016) Augmenting live coding with evolved patterns. *EvoMusArt, Porto, Portugal*, LNCS 9596:31–46. Springer.
16. Hindré T., Knibbe C., Beslon G., Schneider D. (2012) New insights into bacterial adaptation through in vivo and in silico experimental evolution, *Nature Reviews Microbiology*, 10:352–365.
17. Hooper L. V., Gordon J. I. (2001). Commensal host-bacterial relationships in the gut. *Science*, 292:1115–1118.
18. Hooper L. V., Midtvedt T., Gordon J. I. (2002) How host-microbial interactions shape the nutrient environment of the mammalian intestine. *Annual review of nutrition*, 22(1):283–307
19. Hoverd T., Stepney S. (2016) EvoMachina: a novel evolutionary algorithm inspired by bacterial genome reorganisation, *2nd EvoEvo Workshop, CCS 2016, Amsterdam, Netherlands*.
20. Kimura M. (1968) Evolutionary Rate at the Molecular Level. *Nature*, 217:624–626
21. Kriegel H.-P., Krger P., Zimek A. (2009) Clustering highdimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering. *ACM Transactions on Knowledge Discovery from Data*, 3(1):1–58
22. Pagie L., Hogeweg P. (1997) Evolutionary consequences of coevolving targets. *Evolutionary computation*, 5(4):401–418
23. Patrikainen A., Meila M. (2006) Comparing subspace clusterings. *IEEE Transactions on Knowledge and Data Engineering*, 18(7):902–916
24. Peignier S., Rigotti C., Beslon, G. (2015) Subspace clustering using evolvable genome structure. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, pp. 575–582
25. Peignier S., Abernot J., Rigotti C., Beslon, G. (2017) EvoMove: Evolutionary-based living musical companion. In *European Conference on Artificial Life (ECAL)*, pp. 340–347
26. Peignier S., Rigotti C., Rossi A., Beslon G. (2018) Weight-based search to find clusters around medians in subspaces. *ACM Symposium on Applied Computing*. p.10.
27. Plucain J., Hindr T., Le Gac M., Tenaillon O., Cruveiller S., Mdigue C., Leiby N., Harcombe W. R., Marx C. J., Lenski R. E., Schneider D. (2014) Epistasis and allele specificity in the emergence of a stable polymorphism in Escherichia coli. *Science*, 343:1366–1369
28. Rocabert C., Knibbe C., Consuegra J., Schneider D., Beslon G. (2017) Beware batch culture: Seasonality and niche construction predicted to favor bacterial adaptive diversification. *PLoS computational biology*, 13(3):e1005459
29. Rutten J., Hogeweg P., Beslon G. (*in prep*) Adapting the engine to the fuel: mutator populations can reduce the mutational load by reorganizing their genome structure. *in prep*.
30. Szathmry E., Maynard-Smith J. (1997) The major evolutionary transitions, *Nature*, 374(6519):227–232
31. van Dijk B., Hogeweg P. (2016) In silico gene-level evolution explains microbial population diversity through differential gene mobility. *Genome biology and evolution*, 8(1):176–188